Squaring in Reversible Logic using Zero Garbage and Reduced Ancillary inputs

Arindam Banerjee
Dept. of ECE
JIS College of Engineering
West Bengal, India
Email: banerjee.arindam1@gmail.com

Debesh Kumar Das
Dept. of CSE
Jadavpur University
West Bengal, India
Email: debeshd@hotmail.com

***Abstract*—In this paper, our objective is to design an efficient dedicated squaring architecture in reversible logic using reduced ancillary inputs and zero garbage outputs. The proposed design has a modular structure and has been built recursively. The quantum cost optimization has been achieved using $NCV-|1>$ and $NCV-|v_1>$ and double gate libraries. The performance parameters like ancillary inputs, garbage outputs and quantum cost have been formulated from the design and compared with a very recent work [1].**

Keywords : Squarer, Recursion, Reversible Circuits, Garbage, Ancillary Inputs

I. Introduction

Digital arithmetic circuits like multipliers, dividers, squarer etc. are the core components of all the DSP processors, image processing and cryptography. Like other type of arithmetic operations, squaring computation is also used as an important architecture in many applications such as image compression, multimedia applications, cryptography etc. Reversible implementation of squaring circuits can easily be obtained using well known multiplication algorithms [2], [3], [4]. But the main disadvantage of these techniques are the generation of redundant bits. As a result of that the circuit complexity increases drastically. In a recent work [1], appeared in VLSI-2014, the authors have designed a squaring algorithm in reversible logic that avoids the generation of several partial products. In this paper we propose a new technique of squaring with less circuit complexity. Our method also avoids the generation of redundant bits. In our proposed method, the design has zero garbage in the sense that the output of the circuit provides both squarer result and also the input vector. As the input is available at the output, it can be reused for further application. The n bit squarer is designed recursively by appending some gates with the $(n-1)$ bit squarer. These additional gates are performing the multiplication of the most significant bit of the number with the other bits and then the summation of two n bit numbers is done. Here an adder design is proposed where only one ancillary input is required and no other garbage. Our special adder design with single ancillary input has a special advantage in the design of squarer. We show that while used in squarer design some gates can be removed from this adder.

II. Basic Reversible Gates and Circuits

A Boolean function is reversible if and only if it has the same number of inputs and outputs and it maps each input pattern to a unique output pattern. Otherwise, the function is termed irreversible. A reversible function can be realized by a circuit which consists of a cascade of reversible gates. Fan-out and feedback are not allowed [5] in reversible circuits.

Several reversible gates are shown in Fig. 1. Fig. 1(a) is a NOT gate. Fig. 1(b) is a CNOT gate which has two inputs and two outputs. Figs. 1(c), 1.(d) and 1.(e) are 2, 3 and 4 controls Toffoli gates.

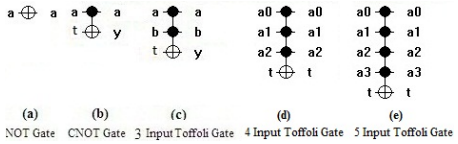

Fig. 1. Basic Reversible gates

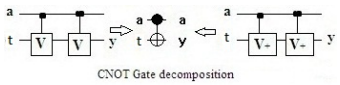

Fig. 2. Decomposition of CNOT gate into V and V+ gate

To compute the quantum cost, circuit decomposition into several elementary gates is essential. In Fig.-2, the CNOT gate decomposition technique has been shown where it is decomposed into two V or V^+ gates.

In our paper, the whole design has been implemented using multiple control Toffoli gates. Therefore a brief description on Toffoli gates is given here.

Description of multiple control Toffoli networks:

A generalized Toffoli gate has a set of control inputs C, a target input set T and has the form $TOF(C,T)$, where $C=(x_{i1},x_{i2},...,x_{ik})$, $T=(x_j)$ and $C\cap T=\emptyset$ [6]. It maps an input vector $(x_1^0,x_2^0,...,x_{j-1}^0,x_j^0,x_{j+1}^0,...,x_n^0)$ to $(x_1^0,x_2^0,...,x_{j-1}^0,x_j^0\oplus(x_{i1}x_{i2}...x_{ik}),x_{j+1}^0,...,x_n^0)$. Thus a NOT gate is $(TOF(x_j))$, a generalized Toffoli gate which has no control. The CNOT gate $(TOF(x_i,x_j))$, which is a generalized Toffoli gate with one control bit, is also known as

978-1-4673-8700-2/16 $31.00 © 2016 IEEE
DOI 10.1109/VLSID.2016.112

Feynman gate. The simple (3×3) Toffoli gate is a generalized Toffoli gate with two controls. An $n\times n$ Toffoli gate has $(n-1)$ control lines and one target line. Therefore this gate is called $(n-1)$ controlled Toffoli gate. Similarly $(n+1)\times(n+1)$ Toffoli gate has n control lines and is called n controlled Toffoli gate. Figs. 1(d) and 1(e) are 3 and 4 controlled Toffoli gates respectively.

The performance parameters for reversible circuit synthesis are defined as follows.

Ancillary Input : It is defined as the constant input required to achieve logical reversibility.

Garbage Output : It is the output other than primary output. It is the extra output to maintain logical reversibility.

Suppose we want to synthesize OR function. The reversible circuit for this is given in Fig. 4 where ancillary inputs and garbage outputs are shown.

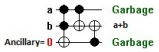

Fig. 3. OR gate describing ancillary inputs and garbage outputs

Quantum Cost : It is defined as the number of elementary operations in a reversible circuit.

To calculate quantum cost of a reversible circuit, the circuit is decomposed into several elementary gates of unit quantum cost as shown in Fig. 2 and the quantum cost is the total number of elementary gates required in the circuit. We are measuring the quantum cost following the method described in [7] where decomposition of n-controlled Toffoli gate has been shown using NCV-$|1>$, NCV-$|v_1>$ and double gate libraries. Following this technique shown in [7], the quantum cost of a n-controlled Toffoli gate is $(2\times n+1)$. For the circuit shown in Fig. 5(a), the calculated quantum cost following this library is 10.

But there are some quantum gates connected in cascade called template whose cumulative operation signifies an identity operation. We can easily reduce the quantum cost of a circuit by searching for such templates and eliminate them. Two cascaded Toffoli gates of Fig. 5(a) can be decomposed as shown in Fig. 5(b). Here two toffoli gates have two control lines in common and at these two control lines there are one V+ gate, one V gate, one controlled V+ gate and one controlled V gate. The rectangular box shown in Fig. 5(b) is a template. This block represents an identity gate and can be eliminated. Thus the total cost becomes 6.

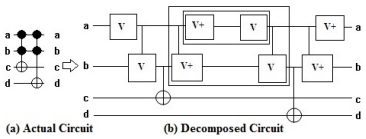

Fig. 4. Decomposition of two cascaded Toffoli gates and cost reduction using template elimination

III. Mathematical Modelling of Squaring Technique

Let A_n is a n bit binary number whose square is to be determined. A_n can be expressed as,

$$A_n = \sum_{i=0}^{n-1} a_i 2^i = a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i = a_{n-1}2^{n-1} + A_{n-1} \tag{1}$$

where, $A_{n-1} = \sum_{i=0}^{n-2} a_i 2^i$. Thus, A_n^2 can be expressed as,

$$A_n^2 = (a_{n-1}2^{n-1} + A_{n-1})^2 \tag{2}$$

$$= a_{n-1}2^{2(n-1)} + 2^n a_{n-1}A_{n-1} + A_{n-1}^2 \tag{3}$$

$$= (a_{n-1} + a_{n-1}a_{n-2})2^{2n-2} + a_{n-1}\sum_{i=0}^{n-3} a_i 2^{n+i} + A_{n-1}^2 \tag{4}$$

Suppose squaring of A is denoted as a $2n$ bit number (S_{2n-1}^n, S_{2n-2}^n, ..., S_1^n, S_0^n). So, A_{n-1}^2 can be expressed as,

$$A_{n-1}^2 = \sum_{i=0}^{2n-3} S_i^{n-1} 2^i \tag{5}$$

where S_i^n is the result of the i^{th} position of A_n^2. Now equation (4) can be rewritten as,

$$A_n^2 = (a_{n-1} + a_{n-1}a_{n-2})2^{2n-2} + Sum_n \tag{6}$$

where, $Sum_n = \sum_{i=3}^{n}(a_{n-1}a_{n-i} + S_{2n-i}^{n-1})2^{2n-i} + \sum_{i=0}^{n-1} S_i^{n-1}2^i$. Let A_{n-1}^2 is known, that is, the bits $(S_{2n-3}^{n-1}, S_{2n-4}^{n-1}, ..., S_n^{n-1}, S_{n-1}^{n-1}, ..., S_2^{n-1}, S_1^{n-1}, S_0^{n-1})$ are given. From equation (6), we can now compute A_n^2 from A_{n-1}^2. Obviously, $S_i^n = S_i^{n-1}$ for $0 \le i \le (n-1)$.

IV. Reversible Adder with reduced ancillary inputs

Suppose we intend to add two numbers $(a_{n-1}, a_{n-2}, ..., a_1, a_0)$ and $(b_{n-1}, b_{n-2}, ..., b_1, b_0)$. Let their summation be given as $(m_n, m_{n-1}, ..., m_1, m_0)$. Generally in a serial full adder design, where carry of every stage is fed from previous stage, the number of ancillary inputs required is n. There are several techniques to do the addition with only single ancillary inputs [8], [9], [10]. We use a new technique to perform addition with single ancillary input. Let us now define two terms as $g_i = a_i b_i$ and $p_i = a_i \oplus b_i$ for $(0 \le i \le n-1)$.

$$m_i = p_i \oplus c_i \tag{7}$$

$$c_{i+1} = g_i \oplus p_i c_i \tag{8}$$

Here $c_0 = 0$.

$$c_1 = g_0 = a_0 b_0 \tag{9}$$

$$c_2 = g_1 \oplus p_1 c_1 = a_1 b_1 \oplus (a_1 \oplus b_1)a_0 b_0 \tag{10}$$

$$c_3 = g_2 \oplus p_2 c_2 \tag{11}$$

$$= g_2 \oplus (a_2 \oplus b_2)(a_1 b_1 \oplus (a_1 \oplus b_1)a_0 b_0) \tag{12}$$

$$= a_2 b_2 \oplus (a_2 \oplus b_2)a_1 b_1 \oplus (a_2 \oplus b_2)(a_1 \oplus b_1)a_0 b_0 \tag{13}$$

$$c_4 = g_3 \oplus p_3 c_3 \tag{14}$$

$$= a_3 b_3 \oplus (a_3 \oplus b_3)(a_2 b_2 \oplus (a_2 \oplus b_2)(a_1 b_1 \oplus (a_1 \oplus b_1) a_0 b_0))) \tag{15}$$

$$c_5 = g_4 \oplus p_4 c_4 \tag{16}$$

$$= a_4 b_4 \oplus (a_4 \oplus b_4)(a_3 b_3 \oplus (a_3 \oplus b_3)..(a_1 b_1 \oplus (a_1 \oplus b_1) a_0 b_0))) \tag{17}$$

Similarly for n bit parallel adder, the Boolean expressions of the outputs can be expressed as follows.

$$c_n = g_{n-1} \oplus p_{n-1} c_{n-1} = g_{n-1} \oplus p_{n-1}(g_{n-2} \oplus p_{n-2} c_{n-2}) \tag{18}$$

$$= g_{n-1} \oplus p_{n-1} g_{n-2} \oplus p_{n-1} p_{n-2} g_{n-3} \oplus ... \oplus (p_{n-1} p_{n-2} ... p_1 g_0) \tag{19}$$

$$m_{n-1} = p_{n-1} \oplus (g_{n-2} \oplus p_{n-2} c_{n-2}) \tag{20}$$

$$= p_{n-1} \oplus g_{n-2} \oplus p_{n-2}(g_{n-3} \oplus p_{n-3} c_{n-3}) \tag{21}$$

$$= p_{n-1} \oplus g_{n-2} \oplus p_{n-2} g_{n-3} \oplus ... \oplus (p_{n-2} p_{n-3} ... p_1 g_0) \tag{22}$$

and so on.

Algorithm 1, given below, has been proposed to design a reversible adder using only single ancillary input and zero garbage. Fig. 6 shows a parallel adder of two four bit numbers following this technique.

Data: i,j,n
Input : n,array a_i,b_i $(0 \leq i \leq (n-1))$
Output: All the reversible gates to produce $sum_0, sum_1, ..., sum_n$
Initialization: $sum_n = 0$, $sum_i = b_i$ for $0 \leq i \leq (n-1)$
for $i \leftarrow 1$ **to** n **do**
  for $j \leftarrow 0$ **to** i **do**
    $G_{i,j} =$
    $TOF(a_{n-i}, sum_{n-i}, sum_{n-i+1}, ..., sum_{n-j})$
  end
end

Algorithm 1: Algorithm for Proposed Addition Approach

In this paper, all the adder stages of the proposed squarer design have been implemented using CNOT and multiple controlled TOFFOLI gates. We will show later that ultimately all $G_{i,j}$'s will not be required in the proposed squarer.

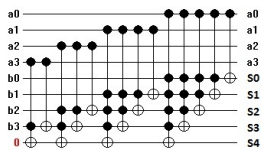

Fig. 5. Architecture of newly proposed 4 bit parallel adder

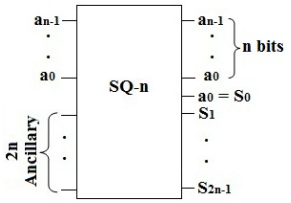

Fig. 6. Schematic Diagram for n bit squaring architecture

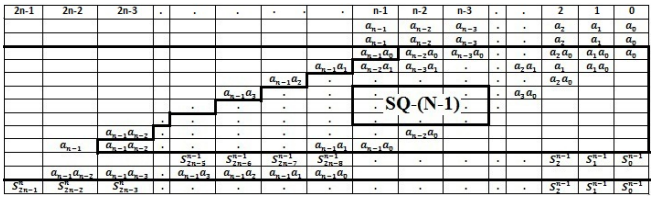

Fig. 7. Schematic Diagram for n bit squaring using $(n-1)$ bit squaring

V. Architecture for squaring methodology

A. Basic Architecture

The proposed squarer of n bits is represented as a block which is shown in Fig. 7. Our architecture is garbage free in the sense that the primary inputs generated as outputs may be used in next iteration or may be used for further computation. The structure consists of $3n$ inputs $[(a_0, a_1, ..., a_{n-1})$ as primary inputs and $2n$ as ancillary inputs] and $3n$ outputs $[(a_0, a_1, ..., a_{n-1})$ outputs same as inputs $(a_0, a_1, ..., a_{n-1})$ that may be used for the next iteration and $(S_0, S_1, S_2, ..., S_{2n-1})$ as primary outputs]. It is to be noticed that though $S_0 = a_0$, still one additional ancillary input has been used to produce S_0 from a_0. Thus at the

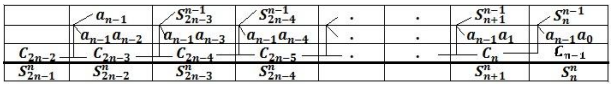

Fig. 8. Schematic Diagram for addition part of higher order n bit

output both the primary inputs and the squared outputs are obtained. We attempt to design the squarer with an iterative fashion to eliminate garbage in each iteration. Let us assume that a squarer unit of a number with $(n-1)$ bits $(a_{n-2}, a_{n-3}, ..., a_1, a_0)$ as primary inputs and its squared output $(S^{n-1}_{2n-3}, S^{n-1}_{2n-4}, ..., S^{n-1}_1, S^0_1)$ is available. In our implementation, we like to achieve the design for n bits using this squarer of $(n-1)$ bits. The schematic diagram for n bit squarer using $(n-1)$ bit squaring architecture is shown in Fig. 8. We can see from Figs. 8 and 9 that the squarer of n bit number can be obtained by adding two $(n-1)$ bit numbers $(a_{n-1}, S^{n-1}_{2n-3}, ..., S^{n-1}_n)$ and $(a_{n-1}a_{n-2}, ..., a_{n-1}a_0)$. Since $(n+1)^{th}$ stage of addition does not exist for $(n-1)$ bit addition then only one carry from the preceding stage must be propagated at the $(n-1)^{th}$ stage to generate the result at the n^{th} position.

In every bit position a carry is generated which is added to

the next bit as observed in Fig. 9. The carries C_{n-1}, C_n,..., C_{2n-2} are shown in Fig. 9. Here it is obvious that $C_{n-1} = 0$ since there is no initial carry. In the column of S^n_{2n-2} shown in Fig. 9, there are three elements to be added (a_{n-1}, $a_{n-1}a_{n-2}$ and C_{2n-3}). The carry generated by a_{n-1} and $a_{n-1}a_{n-2}$ is simply $a_{n-1}a_{n-2}$ which is passed to the next level where it has modulo-2 addition with $(a_{n-1} \oplus a_{n-1}a_{n-2})C_{2n-3}$.

The parallel adder, required to achieve the n bit squared output using $(n-1)$ bit squarer, can be designed in many ways but in this paper we intend to design it by the method described in Section-IV. We perform the generation of partial products and addition simultaneously as described in next subsection, where overall design is described.

B. Overall Design of the squarer

Consider the square of a single bit a_0. Its square is a single bit $S_0 = a_0$ whose implementation is a single line. The square of two bits (a_1, a_0) can be obtained by appending a structure with squarer SQ(1) as shown in Fig. 10. In our design for any n, we are using this basic structure as the first level architecture to be appended with the SQ(n-1) squaring block.

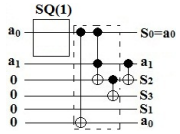

Fig. 9. Architecture of 2 bit squarer using single bit squarer

Now following the tabular interpretation of the architecture shown in Figs. 8 and 9, if we like to design the n bit squarer using $(n-1)$ bit squarer then $(n-1)$ bit parallel adder stage is needed to achieve the exact design. Fig. 11 shows the general architecture for n bit squarer using $(n-1)$ bit squarer. It consists of three parts-(i) SQ(n-1) squaring block, (ii) first level architecture and (iii) modular Toffoli network. This modular Toffoli network along with the first level architecture implements the addition of $(a_{n-1}, S^{n-1}_{2n-3}, ..., S^{n-1}_n)$ and $(a_{n-1}a_{n-2}, ..., a_{n-1}a_0)$. It makes simultaneous implementation of the $(n-1)$ partial products ($a_{n-1}a_{n-2}$,$a_{n-1}a_{n-3}$,...,$a_{n-1}a_0$) and then the summation with $(a_{n-1}, S^{n-1}_{2n-3}, ..., S^{n-1}_n)$. Fig. 12 shows the reversible implementation of Fig. 11. Notice that we get the SQ(2) design of Fig. 10 from SQ(1) design, by adding only the basic structure shown by a dotted box and a CNOT gate. But for $n > 2$, we need to append the basic structure and also a modular Toffoli network. Moreover, from $n = 1$ to $n = 2$, we add four more ancillary inputs, but for $n > 2$, we append only two more ancillary inputs with the structure of $SQ(n-1)$.

Algorithm 2 finds the gates to be appended with the squarer of $(n-1)$ bits to achieve the squarer of n bits. Here $G_{i,j}$ denotes the multiple controlled Toffoli gates and i and j signify the number of levels and gate numbers at level i respectively. The number of control inputs in gate $G_{i,j}$ is $(i-j+1)$ $(1 \le i \le n-1)$ where $(1 \le j \le 3)$ for $i = 1$ and

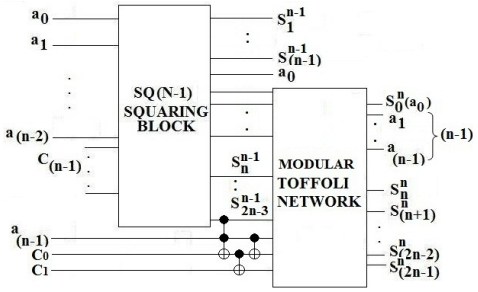

Fig. 10. Schematic Diagram for n bit squaring

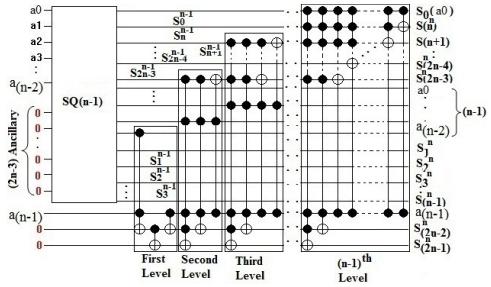

Fig. 11. Reversible implementation for n bit squaring using $n-1$ bit squaring

$(1 \le j \le i+1)$ for $i > 1$. For $i = 1$, there are 3 gates $G_{1,1}$, $G_{1,2}$ and $G_{1,3}$ having 2, 1 and 1 control inputs respectively. The algorithm below represents the implementation of Fig. 12. We next show in the next subsection that some gates in the structure of Fig. 12 can be removed.

Data: i,j,n,a_i for $0 \le i \le (n-1)$, $(1 \le j \le 3)$ for $i = 1$ and $(1 \le j \le i+1)$ for $i > 1$
Input : n,array a_i
Output: Gate array $G_{i,j}$
Initialization: $S^n_{2n-1} = 0$, $S^n_{2n-2} = 0$ and $S^n_i = S^{n-1}_i$
for $(1 \le i \le (2n-3))$ $G_{1,1} = TOF(a_{n-1}, a_{n-2}, S^n_{2n-2})$
$G_{1,2} = TOF(S^n_{2n-2}, S^n_{2n-1})$
$G_{1,3} = TOF(a_{n-1}, S^n_{2n-2})$
for $i \leftarrow 2$ **to** n **do**
  for $j \leftarrow 1$ **to** $i+1$ **do**
    $G_{i,j} = TOF(a_{n-1}, a_{n-i-1}, S^n_{2n-i-1}, ..., S^n_{2n-j})$
  end
end

Algorithm 2: Algorithm for Proposed Squaring

C. Optimization by removing some redundant gates

The squaring mathematics shows us that some gates in Fig. 12 are redundant. As for example for three bit squaring shown in Fig. 13, $S^3_5 = a_2(a_2a_1) = a_2a_1$ which is obtained from the first level circuit. Therefore the Toffoli gate $G_{2,1}$ on the second ancillary input at S^3_5 output position shown in Fig. 13 is redundant and can be eliminated. We propose an efficient gate optimization technique by which the redundant gates can be eliminated.

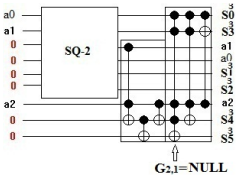

Fig. 12. Schematic Diagram for 3 bit squaring using 2 bit squaring

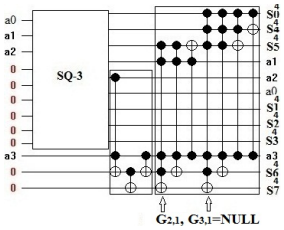

Fig. 13. Schematic Diagram for 4 bit squaring using 3 bit squaring

To achieve this gate optimization for the output S^n_{2n-1}, we first generate a gate array and eliminate the gates accordingly at the positions which are set to 0. Therefore we first introduce few definitions and lemmas essential for the optimization technique as given below.

Definition: Let L_i is defined as the largest value of the pattern $(a_{i-1}, a_{i-2}, ..., a_0)$ such that $L_i^2 < 2^{2i-1}$ and X_i is defined as $(X_i = 2^{2i-1} - L_i^2)$.

The following lemmas are applicable for more than four bit pattern.

Lemma-1: $G_{2,1} = NULL$

Proof: Follows from above discussion. ■

Lemma-2: *If* $L_i < X_i$ *then* $L_{i+1} = 2L_i + 1$

Proof: Let, $Y = 2L_i + 1$ and $Z = 2L_i + 2$. $Y^2 - 2^{2i+1} = 4(L_i - X_i) + 1 < 0$. Now, $Z^2 - 2^{2i+1} = 4(L_i + 1)^2 - 2^{2i+1}$. If $Z^2 - 2^{2i+1} < 0$, then $4(L_i + 1)^2 - 2^{2i+1} < 0$ which means $L_i + 1 < 2^{2i-1}$. It means this L_{i+1} (not L_i) is the largest value such that $(L_i + 1)^2 < 2^{2i-1}$ which is a contradiction. Hence $L_{i+1} = Y_i$. ■

Example: $L_3 = 5$, $X_3 = 7$. It means $L_4 = 11$.

Lemma-3: *If* $L_i > X_i$ *then* $L_{i+1} = 2L_i$

Proof: Let, $Y = 2L_i$ and $Z = 2L_i + 1$. $Y^2 - 2^{2i+1} = -4X_i < 0$. $Z^2 - 2^{2i+1} = 4(L_i - X_i) + 1 > 0$. Hence the proof. ■

Example: $L_4 = 11$, $X_4 = 7$, $L_4 > X_4$. It implies $L_5 = 22$.

Lemma-4: *If* $L_{i+1} = 2L_i + 1$ *then gate* $G_{i-1,1} = NULL$

Example: $L_3 = 5$, $L_4 = 11$. It implies $G_{3,1} = NULL$.

Lemma-5: *If* $L_{i+1} = 2L_i$ *then gate* $G_{i-1,1} \neq NULL$

Example: $L_5 = 2L_4$. It implies $G_{4,1} \neq NULL$.

Lemma-6: *If* $L_i < X_i$ *then the lowest value of* $j \geq 1$ *for which* $L_{i+j} = 2L_{i+j-1}$, *is given by* $j_{min} = \lceil log_2(\frac{1}{L_i - 2^{\frac{2i-1}{2}} + 1}) \rceil$

Proof: As $L_i < X_i$, obviously $L_{i+1} = 2L_i + 1$. Let us assume $K_i = L_i$ and $K_{i+1} = 2K_i + 1$, $K_{i+2} = 2K_{i+1} + 1$, ... $K_{i+j} = 2K_{i+j-1} + 1$. It implies $K_{i+j} = 2^j L_i + (2^j - 1)$. As $L_{i+j} = 2L_{i+j-1}$, obviously $K^2_{i+j} > 2^{2i+2j-1}$. It means $2^j L_i + 2^j - 1 > 2^{(i+j)-\frac{1}{2}}$ or $\frac{1}{2^j} < L_i - 2^{\frac{(2i-1)}{2}} + 1$. It implies that $j_{min} = \lceil log_2(\frac{1}{L_i - 2^{\frac{2i-1}{2}} + 1}) \rceil$. ■

Example: For $i = 3$, $L_i = 5$, $X_i = 7$, $j_{min} = \lceil log_2(\frac{1}{5 - 2^{\frac{5}{2}} + 1}) \rceil = 2$. It means $G_{4,1} \neq NULL$.

Lemma-7: *If* $L_i > X_i$ *then the constant value for which* $L_{i+j} = 2L_{i+j-1} + 1$, *is given by* $j_{min} = \lceil log_2(\frac{1}{2^{\frac{2i-1}{2}} - L_i}) \rceil$

Proof: As $L_i > X_i$, obviously $L_{i+1} = 2L_i$. Let us assume $K_i = L_i$ and $K_{i+1} = 2L_i$, $K_{i+2} = 2K_{i+1}$, ... $K_{i+j} = 2K_{i+j-1}$. It implies $K_{i+j} = 2^j L_i$. As $L_{i+j} = 2L_{i+j-1} + 1$, obviously $(2^j L_i + 1)^2 < 2^{2(i+j)-1}$ which implies that $j_{min} = \lceil log_2(\frac{1}{2^{\frac{2i-1}{2}} - L_i}) \rceil$. ■

Example: For $i = 8$, $L_i = 181$, $L_i > X_i$, $j_{min} = \lceil log_2(\frac{1}{2^{\frac{15}{2}} - 181}) \rceil = 6$.

Data: i,j,temp,L_i,X_i,$G_{i,1}$,l,m,n
Input : n
Output: Gates to be removed
Initialization: $Z_{2,1} = 0$,$L_3 = 5$,$X_3 = 7$,$i = 3$
while $i < n$ **do**
  if $L_i \geq X_i$ **then**
    $l = \lceil log_2(\frac{1}{2^{\frac{2i-1}{2}} - L_i}) \rceil$
    $temp = L_i$
    $i = i + l + 1$
    Remove gate $G_{i-1,1}$
    $L_i = temp \times 2^{l+1} + 1$
  end
  $m = \lceil log_2(\frac{1}{L_i - 2^{\frac{2i-1}{2}} + 1}) \rceil$
  $temp = L_i$
  for $j \leftarrow i + 1$ **to** $i + m - 1$ **do**
    Remove gate $G_{i-1,1}$
  end
  $i = i + m - 1$
  $L_i = temp \times 2^{m-1} + 2^m - 1$
end

Algorithm 3: Algorithm for gate elimination technique

Algorithm 3 describes the technique to eliminate the redundant Toffoli gates from the design. If $G_{i-1,1} = 0$ for $(0 \leq i \leq (n-1))$ then the gate $G_{i-1,1}$ is redundant. Thus Algorithm 3 finds the value of i for which $G_{i-1,1} = 0$. Implementation of 6 bit squaring is shown in Fig. 15, in which $G_{2,1}$, $G_{3,1}$ and $G_{5,1}$ can be removed as they are redundant using this algorithm. If we run the above algorithm for $n = 64$, then $G_{i,1}$ will be equal to 0 for $i = 2, 3, 5, 7, 13, 16 - 19, 22, 23, 26, 27, 31 - 35, 37 - 40, 42, 44, 47, 49, 50, 52, 53, 56, 59, 61, 62, 63$.

VI. Result Analysis

The proposed technique has been compared with the recently published work [1]. The comparative results are tabulated here in Table I. In Table I, column 1 represents the number of input variables, columns 2, 4 and 6 represent the ancillary inputs (A.I), garbage output (G.O) and quantum cost (Q.C) respectively for the design of [1], whereas column 3, 5 and 7 represent the A.I, G.O and Q.C respectively for our

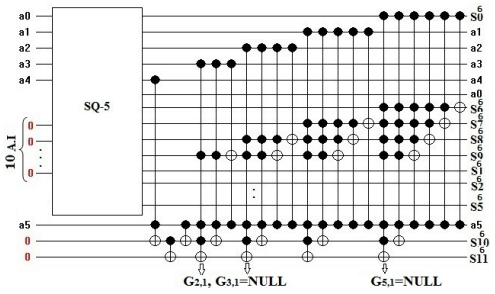

Fig. 14. Implementation of 6 bit squaring using 5 bit squaring

proposed design. In calculating quantum cost, we followed the technique given in [7]. In contrast with the work shown in [1], we have achieved significant improvement in respect of line count. Our design offers garbage free outputs and less number of ancillary inputs as evident from the table.

TABLE I
COMPARATIVE STUDY OF THE PROPOSED ARCHITECTURE WITH RESPECT TO THE DESIGN SHOWN IN VLSI-14 [1]

	A.I		G.O		Q.C	
n	VLSI-14 [1]	Proposed	VLSI-14 [1]	Proposed	VLSI-14 [1]	Proposed
2	3	3	1	0	9	8
3	7	5	4	0	29	23
4	13	7	9	0	60	52
5	21	9	16	0	102	115
6	31	11	25	0	155	201
7	43	13	36	0	219	310
8	57	15	49	0	294	448
9	73	17	64	0	380	615
10	91	19	81	0	477	814
11	111	21	100	0	585	1048
12	133	23	121	0	704	1320
13	157	25	144	0	834	1633
14	183	27	169	0	975	1993
15	211	29	196	0	1127	2400
16	241	31	225	0	1290	2857

VII. CONCLUSION

We have proposed an efficient technique for the reversible implementation of dedicated squarer circuit. Moreover, though the quantum cost of the design is more still it did not go beyond the tolerable limit. The proposed design has modular structure, less ancillary inputs and zero garbage. The zero garbage is considered as the input also appears at the output to be ready for any other computation. However if we consider this input at the output as garbage, then there are only n garbage outputs which is also significantly less. Though our design needs higher quantum cost in general but the achievement is excellent in the sense that it highly decreases the number of ancillary inputs and garbage outputs which require additional pins in the design. In our opinion, decreasing number of ancillary inputs and garbage outputs is more important in the sense that they require additional pins.

REFERENCES

[1] H.V.Jayashree, H.Thapliyal, and V.K.Agrawal, "Design of dedicated reversible quantum circuitry for square computation," in *Proceedings of 27th International Conference on VLSI Design*, January 2014, pp. 551–556.

[2] S.Kotiyal, H.Thapliyal, and N.Ranganathan, "Circuit for reversible quantum multiplier based on binary tree optimizing ancilla and garbage bits," in *Proceedings of 27th International Conference on VLSI Design*, January 2014, pp. 545–550.

[3] M.Haghparast, S.Jassbi, K.Navi, and M.Eshghi, "Optimized reversible multiplier circuits," in *Journal of Circuits, Systems and Computers*, vol. 18, 2009, pp. 311–323.

[4] S.Offermann, R.Wille, G.W.Dueck, and R.Drechsler, "Synthesizing multiplier in reversible logic," in *13th IEEE Symposium on DDECS*, April 2010, pp. 335–340.

[5] M. Nielsen and I. Chuang, "Quantum computation and quantum information," in *Cambridge Univ. Press*, 2000.

[6] H. Rahaman, D. K. Kole, D. K. Das, and B. B. Bhattacharya, "Fault diagnosis in reversible circuits under missing-gate fault model," *Journal of Computers and Electrical Engineering, Elsevier*, vol. 37, no. 4, pp. 475–485, May 2011.

[7] M. Soeken, Z.Sasanian, R. Wille, D. M. Miller, and R. Drechsler, "Optimizing the mapping of reversible circuits to four-valued quantum gate circuits," in *Proceedings of ISMVL-2012*, 2012.

[8] S. Pal, C. Vudadha, S. P. P., S. Veeramachaneni, and S. Mandalika, "A new design of an n-bit reversible arithmetic logic unit," in *ISED-2014*, 2014.

[9] Y. Takahashi, S. Tani, and N. Kunihiro, "Quantum addition circuits and unbounded fan-out," in *arXiv:0910.2530*, October 2009.

[10] S. A. Cuccaro, T. G. Draper, and S. A. Kutin, "A new quantum ripple carry addition circuit," in *arXiv:quant-ph/0410184*, February 2008.